Secure and Efficient k-NN Queries

Hafiz Asif, Jaideep Vaidya, Basit Shafiq, Nabil Adam

Secure and Efficient k-NN Queries*

Hafiz Asif[1], Jaideep Vaidya[1], Basit Shafiq[2], Nabil Adam[1]

[1]MSIS Department, Rutgers University, USA
[2]CS Department, Lahore Univeristy of Managment Sciences, Pakistan

Abstract. Given the morass of available data, ranking and best match queries are often used to find records of interest. As such, k-NN queries, which give the k closest matches to a query point, are of particular interest, and have many applications. We study this problem in the context of the financial sector, wherein an investment portfolio database is queried for matching portfolios. Given the sensitivity of the information involved, our key contribution is to develop a secure k-NN computation protocol that can enable the computation k-NN queries in a distributed multi-party environment while taking domain semantics into account. The experimental results show that the proposed protocols are extremely efficient.

Keywords: k-NN queries, k-NN classification, Privacy, Distributed computation

1 Introduction

Nearest neighbor (NN) queries are an extremely important data analysis tool, and have been used in numerous domains. Indeed, they have been identified (in the form of k-NN classification) as one of the top 10 algorithms in data mining [17], though they can also be used for other applications such as regression, content retrieval, and structure prediction. While the typical use of k-NN does not worry about the sensitivity of the data, k-NN is also applicable in many cases where the data may be private, and the organization interested in querying is different from the organization holding the data.

Consider the financial environment, wherein we have several organizations (such as Ameritrade, Charles Schwab, etc.) which possess financial data about individuals, including their current stock positions, transactional history, etc. Now, a regulating agency such as the SEC may be interested in finding individuals who have a certain stock position, or have indulged in particular type of transactional behavior. Since perfect match is often difficult, best match queries are used to find the closest individuals of interest. Alternatively, a financial advisor service might want to provide recommendations based on similar stock positions or transactional behavior. Typically, since financial data is extremely sensitive, the organizations may be unwilling or even (legally) unable to allow

* The work of Shafiq is supported by HEC grant under the PAK-US Science and Technology Cooperation Program and by HEC NRPU grant. The work of Adam is supported by the National Academies of Sciences, Engineering, and Medicine under the PAK-US Science and Technology Cooperation Program. The work of Vaidya and Asif is supported in part by the National Science Foundation Grant CNS-1422501 and the National Institutes of Health Award R01GM118574. The content is solely the responsibility of the authors and does not necessarily represent the official views of the agencies funding the research.

unfettered access to this data. However, in certain cases, there may be a lot of value associated to be obtained through computing the best match queries. For example, one can identify trading behavior of investors from their portfolio structures as shown by [3]; this kind of information is invaluable for numerous organizations e.g. State Exchange Commission (SEC) in the United States. Similar problems exist in the field of medicine, finance, and homeland security.

In this paper, we address this specific problem. We consider the scenario where several organizations possess independent portfolio databases, each record of which contains financial stock positions for a single portfolio. Together, these databases comprise the global database which contains the portfolios of all entities, though no third party exists which knows this global database in its entirety. Another organization, called the querier, would like to query this global database to retrieve the k portfolios that are the most similar to a particular query portfolio that it possesses. All of the organizations would like to protect the privacy of their information, while still enabling the computation. There has been some work addressing this problem in the past, especially in the context of outsourcing [7, 16]. Our proposed solution improves on the state of the art by providing a way to incorporate the domain semantics and is significantly more efficient. Our solution is also applicable in the outsourcing environment where an organization may want to outsource its database in encrypted form and still enable best match queries. Furthermore, our solution can be extended to provide top-k results based only on private ranking criteria (without reference to a specific query point) in an even more efficient fashion. It is worth noting that while the problem has been formulated in the context of financial domain, our approach is quite general and can be used to solve k-NN and top-k query problems in any domain. Overall, our key contributions are:

1. We introduce the notion of semantic distance which is useful in taking domain semantics into account while computing k-nearest neighbor queries.
2. We propose an extremely efficient multi-party protocol to compute k-NN queries that is robust to semi-honest adversaries.
3. We show how the protocol can be adapted to the outsourced data model and used for k-NN based classification without leaking any additional information.

2 Problem Statement

In this paper, we build a protocol for overcoming the privacy problem for situations where organizations (or people) are interested in finding the best matches for a query over distributed data. As discussed earlier, we formulate the problem in the context of the financial domain i.e., finding investment portfolios from a distributed database, which best match a given query portfolio in a dynamic and semantic aware environment, while also providing confidentiality and security-privacy guarantees to the parties involved in the protocol. We consider a database $\mathcal{P} \in \mathbb{R}^{N \times M}$ is horizontally distributed among n parties $P_1, \ldots, P_n$ such that for all $i \in [n]$ and $N_i \in \mathbb{N}$ the database fragment $\mathcal{P}_i \in \mathbb{R}^{N_i \times M}$ is kept by P_i where $\sum_{i \in [n]} N_i = N$. Each party P_i collects the same features of information but for different entities. These parties could be banks, hedge-funds, mutual-funds, or other institutions. Another party Q is interested in performing a k-NN query, which also incorporates structure and semantics, on this distributed data without revealing its query, while the parties owning data being queried want to avoid disclosure of their data except for the legitimate output of the query. Thus, the problem can be formally defined as follows:

Definition 1 (Distributed Secure k-NN query: DS-kNN). *Given a database $\mathcal{P} \in \mathbb{R}^{N \times M}$, which is horizontally distributed among n parties with party P_i having fragment $\mathcal{P}_i$ $\forall i \in [n]$, a querier Q wants to privately find semantic distance (definition 3) based k-NN in $\mathcal{P}$ to its query $q \in \mathbb{R}^M$ for $N, M, n \in \mathbb{N}$, where $n \geq 2$.*

Definition 2 (Security/Privacy). *A protocol Π computes DS-kNN query securely if it reveals nothing but k-NN records to Q with leakage $\mathcal{L}$ to all other parties with negligible probability in security parameter – a function, $\mu(m)$, is negligible in m if $\mu(m) < \frac{1}{p(m)}$ for all sufficiently large m and polynomial $p(m)$.*

Hence definition 2 ensures that data owners do not learn anything about the query beyond what they already know or infer from leakage, and Q learns nothing about $\mathcal{P}$ beyond what it already knows and can infer from the output and leakage

Adversarial Model: We assume all parties to be non-colluding and *semi-honest* (i.e., honest-but-curious) adversaries, who communicate over a secure channel. However, *restriction on collusion among the data owners can be relaxed to the collusion of at least α data owners*, where $\alpha \leq n$ (total number of data owners). We can accomplish this (without any major change in the proposed protocol) by employing additive homomorphic encryption with threshold α, which for decryption will require α parties each performing partial decryption on an encrypted message.

3 Proposed Approach

We first introduce a notion of semantic awareness for distance metrics that can capture the desired level of granularity and structure for measuring similarity. For example, standard distance metrics such as Euclidean distance fails to capture structural and semantic information such as the industry or sector a stock belongs to, market capitalization, risk and type of the stock, etc. Consider an investor who would like to find similar portfolios, while incorporating portfolio structure and/or commodity relationships with regards to a particular *categorization model* (e.g., the industry classification of stocks [1]). Such a categorization model would typically be built by domain experts. We assimilate the categorization model into the distance metric, denoted `sem_dist`, which we term as the semantic distance. This enables the integration of semantic information representing the true interest of the querier, while evaluating similarity among portfolios (or records). We now formalize the notion of semantic aware distance and then discuss how it will be calculated in a secure (and privacy-preserving) manner.

Definition 3. *Semantic Distance between two points $X, Y \in \mathbb{R}^M$ is Euclidean distance between their linear projection in space $\mathbb{R}^l$, where $l, M \in \mathbb{N}$ and the projection is guided by a given categorization model (map, W), and is formalized as follows.*

$$\texttt{sem\_dist}\,(X, Y) = \sqrt{\sum_{l:c_l \in C} \left(\sum_{j:c_l \in map(s_j)} w_{lj}\,(x_j - y_j) \right)^2}$$

Here, $map : S \rightarrow 2^{\mathcal{C}}$ specifies the category $c_l \in \mathcal{C}$ to which the stock $s_j \in S$ should be mapped with $\mathcal{C}$ and S being set of all the categories and set of all stocks respectively. $C \subseteq \mathcal{C}$ is the set of categories, for which `sem_dist` is to be calculated, whereas w_{lj} in $W = [w_{lj}]_{|\mathcal{C}| \times M}$ gives the number of units of c_l equivalent to one unit of s_j. We also define *signed-distance(X,Y)* at a category level (c_l) to be $D_l = \sum_{j: c_l \in map(s_j)} w_{lj} \, (x_j - y_j)$.

Tuple (map, W) defines categorization model. map gives the relationship among commodities e.g., Industrial Classification Benchmark (ICB) [1] provides a classification for stocks based on the sector and industry; equivalently map could specify the categorization based on market capitalization or some other type. W, here, could denote a weight factor to estimate equivalent worth of a stock in an industry or sector. In general, `sem_dist` allows for a richer query specification, which is very helpful. For example, `sem_dist` allows accounting for risk and/or diversity of each portfolio, while calculating distance between a portfolio and a query portfolio. Here we show, using an example, the effectiveness of semantic distance. Consider Table 1a, which contains three portfolios p_1, p_2 and p_3. Each portfolio specifies the number of stocks of AAV, RDC, ICD, GTT and NOW held in it. The stocks in Table 1a are from Oil and IT sectors. Table 1b gives the conversion factor per share of a stock to equivalent dollar value in a sector. Now, consider an investor who wants to find a portfolio from Table 1a, which is the closest in terms of its value at sector level. If Euclidean distance is directly used, then the results are not meaningful, as can be seen from the results in Table 1c; i.e., according to Euclidean distance, p_2 and p_3 are equally close (similar) to p_1, whereas we notice that in term of the value of portfolio at sector level, p_1 should be closer to p_2 as compared to p_3, since in contrast to p_3, where only 15 shares of stocks are held in IT sector, p_1 and p_2 both hold 10 shares of stocks in Oil sector and 5 shares of stocks in IT sector. On the other hand if we use semantic distance then the results corroborate with our intuition and the semantic meaning of the query asked by the investor; this can be seen from the calculated distances in Table 1d. Though the calculated Euclidean distance is spatially correct, it fails to capture a lot of domain, structural and semantic information.

	AAV	RDC	ICD	GTT	NOW
p_1	0	10	0	5	0
p_2	5	0	5	0	5
p_3	0	0	0	15	0

(a) Portfolios

	AAV	RDC	ICD	GTT	NOW
OIL	80	30	50	0	0
IT	0	0	0	100	50

(b) Weights for Categories

	p_1	p_2	p_3
p_1	0	$10\sqrt{2}$	$10\sqrt{2}$
p_2	$10\sqrt{2}$	0	$10\sqrt{3}$
p_3	$10\sqrt{2}$	$10\sqrt{3}$	0

(c) Euclid distance

	p_1	p_2	p_3
p_1	0	$50\sqrt{74}$	$100\sqrt{109}$
p_2	$50\sqrt{74}$	0	$50\sqrt{794}$
p_3	$100\sqrt{109}$	$50\sqrt{794}$	0

(d) Semantic Distance

Table 1: Example illustrating Semantic Distance Effectiveness

We stress that the proposed protocol is also able to calculate secure DS-kNN query based on simple Euclidean distance measure. In semantics distance formulation, this can be accomplished by a $map : S \to \mathcal{C}$, which is bijective, and setting W to identity matrix of dimension $|S| \times |S|$. This will essentially calculate the Euclidean distance between two points.

3.1 DS-kNN Query Protocol

Before presenting the details of the approach to compute the k closest portfolios as per the problem definition above, we first present the underlying assumptions, the notation used, and a few preliminaries. N, S and $\mathcal{C}$ are known to all the parties including querier. Furthermore, each data owner (P_i) knows the size ($N_j \ \forall j \in [n]$) of all databases fragments $\mathcal{P}_j$. The database can be viewed as a matrix. In rest of the paper by parties/party we mean parties/party owning data, whereas Q is referred to as querier. Additive homomorphic encryption (AHE) e.g., Paillier allows addition of two encrypted values and multiplication of encrypted value with a plain-text value. The plain text values on which AHE operates come from $\mathbb{N}$; let us say for a given security parameter λ AHE accepts plain-text values from $P(\lambda)$ such that $\forall x \in P(\lambda),\ x < \Lambda$, where $\Lambda \in \mathbb{N}$. We divide $P(\lambda)$ into two halves where lower half is positive and upper half is negative (i.e. contains additive inverses of lower half). Whenever a text is decrypted it is converted to equivalent negative or positive value; additive inverse of an encrypted value x i.e., $E_{pk}[x]$, is $E_{pk}[x]^{\Lambda-1}$. As for the decimal values we can decide for a precision up to a decimal point d, then multiply each plain-text value with 10^d and convert it to an integer value. $\lambda \in \mathbb{N}$ is picked in such a manner that the finally computed plain-text value in encrypted form is always within the range. We also employ garbled circuit [19] for secure comparison.

The basic idea in *DS-kNN* is for data owners to encrypt the portfolio database and send it to Q, who calculates *signed-distances* w.r.t. its query, q, in encrypted form (Algorithm 1), and uses them to collaboratively calculate semantic distances in form of random shares with a data owner (Algorithm 1). A distributed rank query is then carried out to identify the indices of k portfolios with the smallest distances (Algorithm 2). Finally, Q retrieves portfolios, corresponding to the indices identified above, from the portfolio database.

We now discuss the details. In Algorithm 1 we outline the algorithm for DS-kNN, where a party P_t is picked at random from data owners to initiate the protocol. P_t can be picked by each party generating a random number r_i from $[n]$ and then calculating $\sum_{i\in[n]} r_i \mod (n+1)$ using secure sum [6]. P_t generates public-private (pk, sk) key pair of AHE and sends it to all data owners and pk to Q. P_t also picks two parties P_l and $P_{l'}$ randomly and lets all the parties know who they are. Next, every P_i first permutes its database, and creates encrypted shares $EP_i = E_{pk}[\mathcal{P}_i^{\pi_i} - \mathcal{R}_i]$ and $ER_i = E_{pk}[\mathcal{R}_i]$. Note that these are homomorphically encrypted additive random shares of $\mathcal{P}_i$. Now P_i sends EP_i to P_l and ER_i to $P_{l'}$. This ensures that every database is split into two parts and thus prevents the leakage of any information to other parties or to Q. P_l and $P_{l'}$ put all of these shares together and permute them using a common random seed $\tilde{s}$ (which can be done by having P_l and $P_{l'}$ each pick a random number and send it to the other and then compute the XOR of both random numbers). These encrypted permuted

Algorithm 1 DS-kNN

```
**Input:** $(m, \mathcal{P}_i)\ at\ P_i\ \forall[n]$, security parameter and database of portfolios
**Input:** At $Q$: $q$, query; $(map, W)$, categorization model; $k$, number of NN; $m$
**Output:** $Q$ gets k-NN portfolios
Generate random seed, $s_i,\ at\ P_i\ \forall i \in [n]$
All data owners, $P_1, \ldots, P_n$, together pick $t$ uniformly from $[n]$
$P_t$ generates $(pk, sk)$, key pair, for AHE and shares it with all data owners, and $pk$ with $Q$
$P_t$ picks $l$ from $[n]$ and $l'$ from $[n] \setminus \{l\}$ uniformly and sends $(l, l')$ to all data owners
**for** each $P_i\ \forall i \in [n]$ **do**
    Generate a matrix $\mathcal{R}_i$ of random numbers: $\mathcal{R}_i \leftarrow \mathbb{N}^{N_i \times |S|}$$\{N_i$ is the size of $\mathcal{P}_i\}$
    Permute the database $\mathcal{P}_i$: $\mathcal{P}_i^{\pi_i} = \pi(s_i, \mathcal{P}_i)$
    Create encrypted random shares of $\mathcal{P}_i^{\pi_i}$: $(EP_i, ER_i) = (EP_{pk}[\mathcal{P}_i^{\pi_i} - \mathcal{R}_i],\ EP_{pk}[\mathcal{R}_i])$
    Send $EP_i$ to $P_l$ and $ER_i$ to $P_{l'}$
$P_l$ sets $EP = \pi(\tilde{s}, (EP_1, \ldots, EP_n))$$\{$seed $\tilde{s}$ is picked together by $P_l$ and $P_{l'}\}$
$P_{l'}$ sets $ER = \pi(\tilde{s}, (ER_1, \ldots, ER_n))$
$P_l$ and $P_{l'}$ respectively send $EP$ and $ER$ along with $t$ to $Q$
$Q$ permutes $EP$ and $ER$ using random seed $\hat{s}$: $(EP^{\pi}, ER^{\pi}) = (\pi(\hat{s}, EP), \pi(\hat{s}, ER))$
$Q$ sets $q_{enc} = E_{pk}[q]^{(\Lambda - 1)}$
$Q$ initializes matrices, $T$ and $D_Q$, of sizes $N \times |C|$ and $N \times 1$, to have $E_{pk}[0]$'s and 0's resp.
$P_t$ initializes $D_t$ as matrix of 0's with size $N \times 1$
**for** each $i \in [N]$ **do**
    **for** $j \in \{1, \ldots, |S|\}$ **do**
        $Q$ sets $T[i, map[j]] = T[i, map[j]] \times (EP^{\pi}[i][j] \times ER^{\pi}[i][j] \times q_{enc}[j])^{W[map[j],j]}$
    **for** $l \in \{1 \ldots |C|\}$ **do**
        $Q$ generates random numbers $r$ and $g$ and sets $T_Q[i, l] = -r^2 + g$
        $Q$ sets $v_r = E_{pk}[T[i, l]] \cdot E_{pk}[r]^{(\Lambda - 1)}$ and $v_g = E_{pk}[T[i, l]]^{2r} \cdot E_{pk}(g)^{(\Lambda - 1)}$
        $Q$ sends $(v_r, v_g)$ to $P_t$
        $P_t$ sets $T_t[i, l] = D_{sk}[v_r]^2 + D_{sk}[v_g]$$\{D_{sk}$ decrypts to equivalent +ive/-ive value$\}$
        $Q$ sets $D_Q[i] = D_Q[i] + T_Q[i, l]$
        $P_t$ sets $D_t[i] = D_t[i] + T_t[i, l]$
$Q$ and $P_t$ interactively find indices of $k$-smallest distances: $I \leftarrow \texttt{k-Smallest}(D_Q, D_t, k)$
**return** portfolios corresponding to $I$ to $Q$ by getting their random shares decrypted from $P_t$
```

shares (denoted EP and ER) are then sent to Q. At this point, Q also randomly permutes the received shares of the database to avoid linkage attack by data owners. Q can reconstruct the database by adding together the received shares of database in encrypted form. Q then proceeds to calculate *signed-distance* in encrypted form according to its specified (map, W) and C. This steps consists of addition of encrypted values and their multiplication by values in plain-text (weights), which can be done in encrypted form thanks to the additive homomorphism.

Next, these signed distances need to be squared. This is accomplished by Q generating random numbers, r and g for each signed-distance, $v_{i,l}$. Q sets its random share for the squared signed-distance to be $v_{i,l}^1 = -r^2 + g$, and sends P_t, $E_{pk}[v_{i,l} - r]$ and $E_{pk}[2rv_{i,l} - g]$. P_t decrypts the received encrypted messages, converts them to appropriate negative or positive values as explained in preliminaries and sets its share to be $v_{i,l}^2 = (v_{i,l} - r)^2 + 2rv_{i,l} - g$. It is obvious that $(v_{i,l})^2 = v_{i,l}^1 + v_{i,l}^2$. Summing all

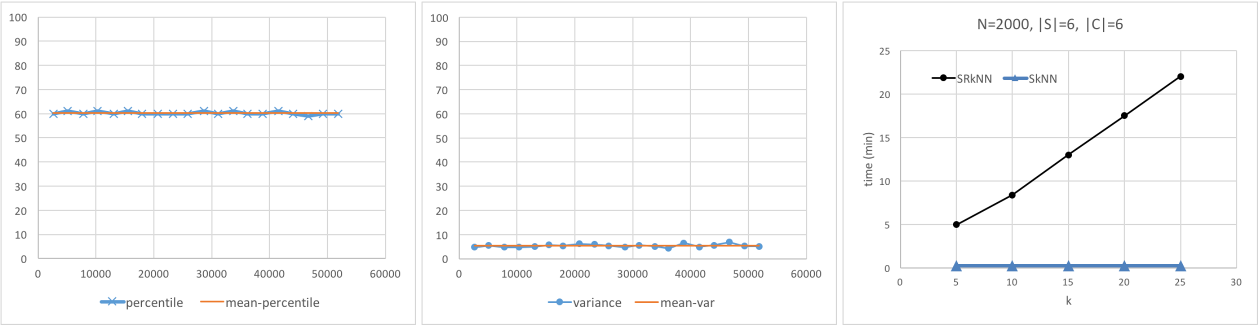
Algorithm 2 k-Smallest(D_Q, D_t, k)

```
Input: At $Q$: $D_Q$, at $P_t$: $D_t$, such that $D_Q$ and $D_t$ are random shares of the squared distances
Input: At $Q, P_t$: $k$, the number of closest records desired
Output: At $Q$: $I$, the array containing indices of smallest elements in $D_Q + D_t$
At $Q$: $\forall i, V_Q[i] = D_Q[i] \times |D_Q|$
At $P_t$: $\forall i, V_t[i] = D_t[i] \times |D_t| + i$
while $k > 0$ do
    $Q$ sets $(\mu_Q, \ell) = (mean(V_Q), |V_Q|)$, and $P_t$ sets $(\mu_t, \ell) = (mean(V_t), |V_t|)$
    for $i \in 1 \dots \ell$ do
        $Q$ sets $U_Q[i] = (V_Q[i] - \lfloor \mu_Q \rfloor)$; $P_t$ sets $U_t[i] = (-V_t[i] + \lfloor \mu_t \rfloor)$
        if $U_Q[i] \le U_t[i]$ {At $Q, P_t$: Yao Comparison} then
            Remove $i^{th}$ element from $V_Q$ and $V_t$ and add to $V'_Q$ and $V'_t$ at $Q$ and $P_t$ respectively
    for $j \in \{Q, t\}$ do
        if $|V'_j| > k$ then
            set $V_j = V'_j$
        else
            For each element in $V'_j$, add to $I$ the index of corresponding element in $D_j$
            $k = k - |V'_j|$
return $I$
```

shares of squared signed-distances of a portfolio will give the share of square of semantic distance for the portfolio. Thus Q and P_t can compute their shares for the distance for each portfolio since $\texttt{sem\_dist}(p_i, q)^2 = \sum_{l=1}^{|C|} v_{i,l}^1 + \sum_{l=1}^{|C|} v_{i,l}^2$. At this point, the square of semantic distance between each portfolio and the query has been randomly split between Q and P_t. For the sake of efficiency we do not compute the square root of squared semantic distance. However, this does not impact correctness of the protocol. Henceforth, Q and P_t engage in an interactive protocol to compute the k smallest distances corresponding to k-NN portfolios. To find the k-smallest entries from the split distance vectors we develop a novel protocol `k-Smallest` that can accomplish this both securely and efficiently. We first present the simple-k-smallest (SKS) protocol, that efficiently computes the k-smallest entries without worrying about security. For a given a vector V containing *unique* values, v, and $\mathcal{S} = \{\}$, the k-smallest values can be found as follows:

- **1:** Set $\mu = \sum_{v \in V} \frac{v}{|V|}$ and divide V into $V_g = \{v \in V : v > \mu\}$ and $V_{le} = V \setminus V_g$
- **2:** If $|V_{le}| > k$, set $V = V_{le}$ and go to step 1
- **3:** If $|V_{le}| \le k$ then set $V = V_g$, $\mathcal{S} = \mathcal{S} \cup V_{le}$ and $k = k - |V_{le}|$
- **4:** if $k \neq 0$ go to step 1, terminates otherwise

SKS terminates, since each iteration reduces size of V. Note that, only the correct distances are added to the output in step 3 since the distances in V_{le} are guaranteed to be smaller than the ones in V_g. SKS works very well for our problem setting and can easily be extended to be secure. *Though any point in V instead of arithmetic mean can be used to split V without affecting correctness of the algorithm, choice of arithmetic mean as a split point is quite effective as long as subsets (of different sizes) of data are not highly skewed to the left for small values of k.* This assumption does hold in real world



(a) mean distance's percentile

(b) percentile's variance

(c) Comparison with Samanthula et al. [7]

Fig. 1: Empirical analysis

data. Specifically, we show through empirical analysis that portfolio distances for real world stock market data [3] are but slightly skewed to left. We used portfolio data of hundred thousand individuals, which was collected over the period of three years from Swedish stock market [3]. We calculated the mean distance and variance for the mean distance over samples of various sizes (i.e., number of portfolios). Figure-1a depicts the percentile for mean distance and average percentiles for mean distances. It can be seen that mean distance is consistently at percentile 60. Figure-1b depicts the variance for the above calculated percentile for mean distance, and the average variance, which asserts that percentile for mean distance does not vary much. The complexity of SKS for such a distribution will be $O(|V|)$ regardless of value for k. *In the case where data is highly skewed or follows exponential distribution or leakage function is different a randomly picked data points can be used as a split point instead.*

Now we focus on devising a secure and distributed SKS so that it can be carried out on random shares of distances without violating the privacy. It is easy to see that if the first step of SKS can be performed in a secure and distributed form (note, in our case, D_Q and D_t together give V), the remaining steps can be performed locally at Q and P_t.

SKS requires the distances in vector to be *unique* i.e. $\forall i \in [N]$, $D_Q[i] + D_t[i]$ is *unique*. This is necessary, not only to guarantee that the protocol terminates but also to ensure security. In essence, if distances were non-unique, it could have been possible that all of the distances were same, thus resulting in $V_{le} = V$ for all iterations. Since, in our case, uniqueness does not generally hold; therefore, we use a perturbation mechanism to achieve uniqueness. This can be accomplished by scaling the distance of portfolio p_i by N, and translating it by i. In the protocol, `k-smallest`, it is carried out as follows: Q multiplies $D_Q[i]$ with N while P_t multiplies $D_t[i]$ with N and adds i to it (lines 1-2 of algorithm 2) (Note: $|D_Q| = |D_t| = |\mathcal{P}| = N$), which will together gives us $N \times (D_Q[i] + D_t[i]) + i$. Next we need to devise a secure and distributed protocol to compute mean and identify indices of D_Q or D_t for which distances $(D_Q[i] + D_t[i])$ are greater than the mean. If we let mean distance to be $\mu = \mu_Q + \mu_t$, where $\mu_Q = \sum_i D_Q[i]/N$ and $\mu_t = \sum_i D_t[i]/N$ then

$$D_Q[i] + D_t[i] > \mu \iff D_Q[i] - \mu_Q > -D_t[i] + \mu_Q$$

The above observation tells us that result of comparing the distance for a record against the mean distance can be equivalently obtained by comparing the difference between random share and the mean of random shares. Note that since Q and P_t can locally compute this difference, the parties can simply use a secure comparison [19] (the garbled circuit approach) to compute the first step of SKS in a secure manner. Furthermore, Since we are using a finite integer field, it is possible that μ_Q and μ_t are fractional, and hence outside the field. To avoid this we use the output of floor-function on μ_Q and μ_t and employ the following comparison instead: $D_Q[i] - \lfloor \mu_Q \rfloor > -D_t[i] + \lfloor \mu_t \rfloor$, but it does not affect the performance of algorithm 2 since $0 \le \mu - (\lfloor \mu_Q \rfloor + \lfloor \mu_t \rfloor) < 2$. Since the remaining steps are local, both parties can calculate k-smallest entries securely and identify their corresponding indices in D_Q or D_t.
Result Extraction: Using above found indices Q can identify and obtain k-NN portfolios from the encrypted database. Let I contain the indices of k-smallest distances and $\forall j \in I,\ \hat{t}_j$ and $\tilde{t}_j$ be the corresponding records in EP and ER then for all j, Q asks P_t for decryption of $E_{pk}[\varphi_j] = E_{pk}[\hat{t}_j] \otimes E_{pk}[\tilde{t}_j] \otimes \gamma_j$, where γ_j is uniformly picked vector of size $|S|$ from an appropriate domain and $\otimes$ gives coordinate-wise product of two vectors. It is straightforward to compute $E_{pk}[\varphi_j]$ for homomorphically encrypted values and the original record t_j from φ_j i.e., $t_j = \varphi_j - \gamma_j$. Thus completing the protocol for computing `sem_dist` based k-NN for horizontally fragmented database in a privacy preserving fashion.

3.2 Extensions

We can easily extend the protocol devised above to work for the outsourced data model. It can also be used for k-NN classification. Both of these are briefly described below.
Outsourcing Case: Our protocol can very simply be applied for the case where the computation of data owners is transferred to the cloud in a secure manner. Parties can pick non-colluding, semi-honest and untrusted servers C_1 and C_2 to take responsibilities of P_l and $P_{l'}$ respectively except for creation random shares of their databases and their encryption. All the responsibilities of P_t for distance and k-smallest computation along with decryption for result retrieval phase are handed to one of the servers. Once responsibilities have been assigned to C_1 and C_2, following the protocol stated in DS-kNN will compute k-NN securely in cloud.
k-NN Classification Case: The proposed protocol also has the ability to carry out k-NN classification with a very small modification. Let us say there are G classes with labels $\{1, 2, \ldots, G\}$. We append each database fragment with G new columns and name them $1, 2, \ldots, G$. For each row with class label g only column g of the appended G columns will have the value 1, and value 0 for the others. Now all the steps outlined in DS-kNN are carried on the database with appended columns, except for the result retrieval step; furthermore, appended columns are not used for k-NN computations. Once Q has identified k-NN records in encrypted database, it computes a vector $\mathcal{G}$, where $\forall g \in [G],\ \mathcal{G}[g]$

contains k minus the sum of values in column g of k-NN records i.e., k minus the number of votes for each class; thus smaller the value $\mathcal{G}[g]$, higher the number of votes for class g. Next, Q permutes $\mathcal{G}$, creates random shares of values in $\pi(\mathcal{G})$ and send them to P_t, after which both Q and P_t follow `k-smallest` protocol with $k = 1$. At the end of `k-smallest` Q is able to identify the class of its instance q.

4 Complexity Analysis

Let N be the number of portfolios in database $\mathcal{P}$ horizontally distributed among parties $P_1, \ldots P_n$, where each record is of dimension $M = |S|$. The asymptotic computational complexity of DS-kNN is $O(N^2) = O(NM, N \times |C| + N^2)$ since in the worst case there will be $O(NM)$ encryption and $O(N \times |C|)$ decryption operations along with $O(N^2)$ secure comparison by data owners, whereas querier will perform $O(N \times |C|)$ arithmetic operations on encrypted values and $O(N^2)$ secure comparisons; furthermore, in most of the application scenarios $M, |C|, k \ll N$.

It is important to note that in real world data for portfolios require only $O(N)$ comparison to find k smallest entries as shown in Figure 1 and is explained in section 6. Moreover, $|C|$ would also be much smaller as compared to $|S|$ because thousands of stocks are traded in the market. So for all practical purposes asymptotically complexity for our problem will be $O(NM)$ Following the same reasoning as above the asymptotic communication complexity of DS-kNN will also be $O(NM)$.

With respect to the communication complexity, *it may appear that the cost of transferring the entire database over is excessive. While this is true in terms of the communication itself, both the monetary and time cost of doing this is negligible*, since currently available bandwidth and speed are quite high e.g, currently ISPs are providing 1000 Mbps connection to residential users and small businesses, which allows an encrypted database of million rows and ten attributes to be transferred in matter of few seconds. On the other hand, in many cases cost and the time required for secure operations are significantly higher than that of required for data transfer. Additionally, many of the secure protocols including [7, 12] require transferring complete database between/among the parties. Therefore, we believe that this cost is reasonable.

5 Security Analysis

In this section we analyze the security of DS-kNN under the framework of definition 2. We want to show the following:

DS-kNN is secure if probabilistic polynomial time simulators $\mathcal{S}_i(m, \mathcal{P}_i, \mathcal{L}_P)$ and $\mathcal{S}_Q(m, q, (map, W), \mathcal{L}_Q, \mathcal{O}_k)$ can respectively simulate the view of P_i, $\forall i \in [n]$ and Q during the execution of DS-kNN.

This means that if $\mathcal{S}_i$ is provided with P_i's input $(m, \mathcal{P}_i)$ and leakage $\mathcal{L}_P$ (which gives $(J, j_1, \ldots, j_J, N_1, \ldots, N_n, |C|)$), and $\mathcal{S}_Q$ with Q's input $(m, q, (map, W))$ and output ($\mathcal{O}_k$ i.e. k-NN records) along with $\mathcal{L}_Q$ (which gives $(J, j_1, \ldots, j_P, N, S)$) then these simulators will have the same view as their respective parties; thus asserting that DS-kNN reveals no extra information and does fulfill the security definition 2. In the output of leakage functions J is the total number of iterations taken in Algorithm 2 corresponding to $\mathcal{P}$ and q, j_ℓ is the percentile of mean distance in ℓ^{th} iteration, whereas rest of the symbols are same as defined previously.

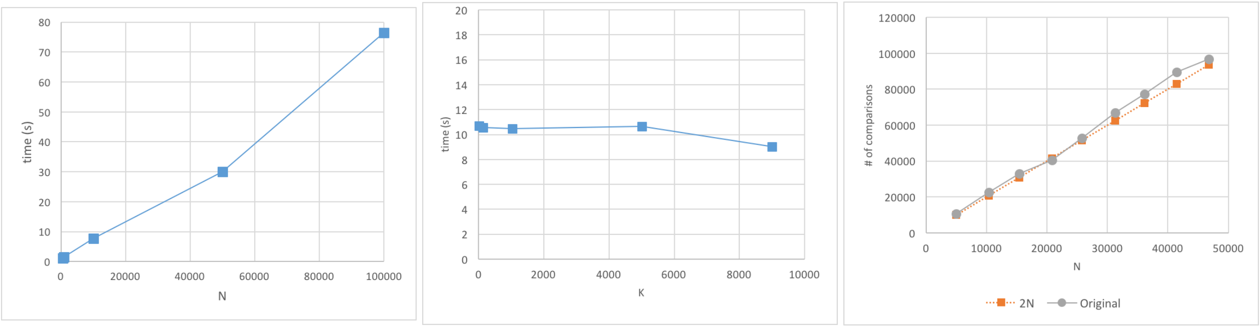
Let us analyze $\mathcal{S}_j$ for the situation where $j = t$ and also $j \in \{l, l'\}$ (l, l' and t are as per specification in Algorithm 1) since such a party, P_j, will receive the biggest set of intermediate messages, in all other cases parties receive less information. P_j's view consists of its input $(m, \mathcal{P}_j)$, random shares $(\mathcal{D}_i \in \mathbb{N}^{N_i \times |S|},\ \forall i \in [n])$ of database fragments, random shares $(\mathcal{H}_R \in \mathbb{N}^{N \times |C|})$ of distances at category level, random seeds $(s_j, \tilde{s})$, random-tape $(r_j \in \{0,1\}^{p(m)})$ and $\mathcal{L}_P$. P_j's view can easily be generated by $\mathcal{S}_j$: based on $m, \mathcal{P}_j$ and $\mathcal{L}_P$ that are provided to $\mathcal{S}_j$, it can generate $\mathcal{D}_i^j \leftarrow \mathbb{N}^{N_i \times |S|}$, $\mathcal{H}_R^j \leftarrow \mathbb{N}^{N \times |C|}$, $(s'_j, \tilde{s}'_j) \leftarrow \mathbb{N}^2$ and $r'_j \leftarrow \{0,1\}^{p(m)}$, $\mathcal{L}_{P_j}$ using uniform distribution. Thus P_j's view, $(m, \mathcal{P}_j, \mathcal{D}_1, \ldots, \mathcal{D}_n, \mathcal{H}_R, s_j, \tilde{s}_j, r_j, \mathcal{L}_P)$, is computationally indistinguishable from $\mathcal{S}_j$'s view, $(m, \mathcal{P}_j, \mathcal{D}_1^j, \ldots, \mathcal{D}_n^j, \mathcal{H}_R^j, s'_j, \tilde{s}'_j, r'_j, \mathcal{L}_P)$, in polynomial time, otherwise pseudo-random generator, which is assumed to be secure, can be broken which is used to create random shares and seeds. It is straightforward that for all other cases a party's view will consist of less information than that of P_j's view; hence $\forall i \in [n],\ \mathcal{S}_i$ will be able to generate P_i's view.

The case for $\mathcal{S}_Q$, is also very similar in that respective inputs, output, and leakage is provided to $\mathcal{S}_Q$, except for the difference that Q receives encrypted database $EP \in \mathbb{N}^{N \times |S|}$ instead of random share of a database, but since AHE is (semantically) secure – meaning EP is computationally indistinguishable from $EP' \leftarrow \mathbb{N}^{N \times |S|}$ (i.e. generated uniformly)– $\mathcal{S}_Q$ can generate a view using $m, q, (map, W)$ and $\mathcal{L}_Q$ that is indistinguishable from Q's view. *Thus proving that DS-kNN is secure.*

Note that the defined leakage reveals information, usually known in our application scenario. If one wants to hide this information then following is one way to accomplish this. Instead of mean distance, randomly picked distances can be used for the purpose of comparison to find k-smallest distances; dummy portfolios with sentinel values can be added to hide size of database; extra columns can be added for dummy coordinates, mapping to which can be provided through a secure and modified bloom filter. $|C|$ can be hidden by adding dummy signed-distances with value zero. Though such measures will stop the leakage, they will significantly reduce the efficiency of the protocol.

6 Experimental Evaluation

We implemented DS-kNN in Java. The platform used for testing is asymmetric in terms of its computational power. The querier machine had a 2.2 GH core-i7 processor and 16 GB RAM whereas each of the database owners was a Xeon E5-2680 v2, with 10 cores running at 2.80GHz, and 96 GB RAM. For AHE and garbled circuit we employed the implementation available at [2] with key size 1024 and [9] with a key size of 512 respectively. The default values for parameters are set based upon domain semantics. Specifically, even though 2k-3k stocks are traded on the stock exchange, only a few hundred of them are most often traded; thus we set $|S|$ to be 100. $|C|$ is set to be 10 because ICB [1] classification taxonomy segregates stocks to 10 industries at the top level. As for k, it is set to 1, which represents the worst case for Algorithm 2. Lastly, N is set to be 1000. For each experiment, only one parameter is varied, while keeping the

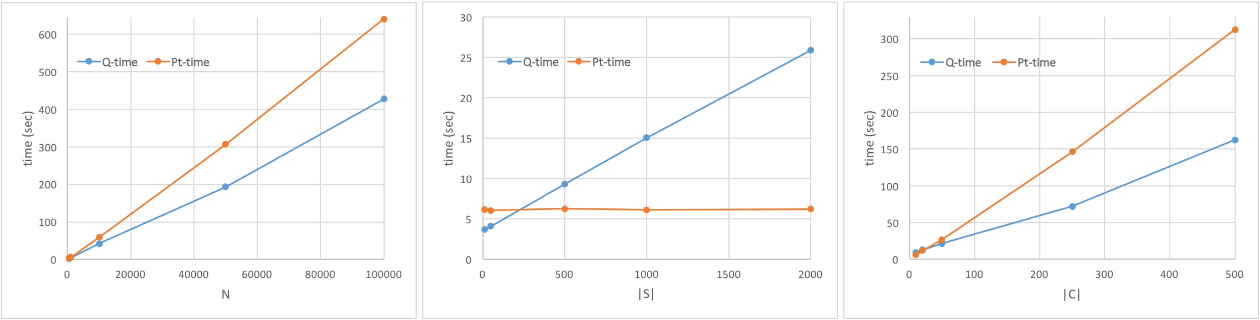


(a) Varying N

(b) Varying k

(c) k-smallest on real data

Fig. 2: k-smallest (Algorithm 2) computation time



(a) Varying N

(b) Varying $|S|$

(c) Varying $|C|$

Fig. 3: Distance computation time for Q and P_t

rest constant. Experiments described below were carried out with synthetic data. The results on real data are described later.

Figures 3a-3c report time taken for distance computation by Q and P_t, with varying N, $|S|$ and $|C|$ respectively. Time for all of these experiments grows linearly except for P_t w.r.t. $|S|$. This is because of the fact that distance computation time for P_t depends on $|C|$ and N, but not on $|S|$. Let us now look at the performance of Algorithm 2, which only depends upon N. Figure 2a plots the computation time taken by Algorithm 2 for varying values of N. Again, the computation time scales linearly w.r.t. N. Figure 2b plots the computation time with respect to varying k. It is interesting to note that the time taken is roughly constant, and thus the time taken by our approach is actually independent of k.

We also compare our work with Elmehdwi et al. [7] for outsourcing case. Figure 1c compares the complete time taken by DS-kNN and SRkNN [7]. For the sake of fair comparison results are computed for same parameters and equivalent processing power. It can be seen that DS-kNN outperforms existing state of the art by an order of magnitude. Additionally, our implementation is in Java and uses threading only for decryption at P_t, whereas implementation in [7] uses the openMP parallelization framework. Thus with an equivalent implementation, our results can be further improved.

Performance on real data: We obtained Swiss stock market data for year 2009-2011, which is a collection of portfolios of around 100k individuals for 300 stocks; the data was previously used in [3]. We only evaluated the performance of `k-smallest` protocol because time for distance computation is independent of data distribution. We randomly picked a subset of the data and choose one portfolio from it as the query portfolio, and computed the number of actual comparisons required by the `k-smallest` protocol for $k = 1$. Figure 2c plots the number of comparisons carried out for different values of N (the number of portfolios) along with a reference line for $2N$. The two lines are almost perfectly in lock-step, which demonstrates the efficiency and suitability of our algorithm for real world data. However, in the worst case, it is still possible that in each iteration only one distance will be removed resulting in $O(N^2)$ total comparisons.

7 Related Work

Privacy-preserving data mining has received a lot of attention [15]. Given the numerous practical applications of privacy-preserving k-NN search, various protocols have been developed to address this problem. [5,14] present solutions to the problem of computing k-NN, where the data is fragmented among different parties, while also preserving privacy. [4] uses a semi-trusted third party to find best k matches. In [12] Qi et al. introduce a single-step protocol for k-NN search, whereas [8] proposed a secure k-NN searching protocol based on PIR for location-based services. However, none of the above work is appropriate for computation over encrypted data. [10] solves recommendation problem using Self-Organizing Map for clustering and k-NN based collaborative filtering, but reveals query to data owner. Zang et. al. in [20] employ homomorphic encryption for finding k-NN in distributed setting, but in contrast to our work it reveals distances, partial access pattern to the parties. In [18], the query along with k-NN distances is exposed and the output is less accurate. [11] makes use of untrusted third party and reveals query to parties. Although semantic distance can be applied here, the *categorization model* will be revealed to data owners. Shaneck et. al [13] provide a solution that reveals partial access pattern while being slower than our proposed protocol. Not only is our protocol straightforwardly extensible to provide outsourcing and k-NN classification, but it also allows for incorporation of semantic distance, while still being comparatively very efficient as compared to state of the art [7].

8 Conclusion and Future Work

In this paper we have presented a secure approach to computing k-nearest neighbor queries for horizontally distributed data. Our approach is an order of magnitude faster than the existing state of the art. It is also applicable in the outsourcing environment, and can be used to compute top-k queries, as well as k-NN based classification. In the future, we plan to develop solutions that are resilient to stronger adversaries, some of which may collude as well.

References

1. Industry classification benchmark. http://www.icbenchmark.com/ICBDocs/Structure_Defs_English.pdf (2015), [Online; accessed 06-Feb-2016]
2. Pailliar encryption implementation. https://code.google.com/p/thep/ (2015)
3. Bohlin, L., Rosvall, M.: Stock portfolio structure of individual investors infers future trading behavior. PloS one 9(7), e103006 (2014)
4. Boneh, D., Gentry, C., Halevi, S., Wang, F., Wu, D.J.: Private database queries using somewhat homomorphic encryption. In: International Conference on Applied Cryptography and Network Security (Jun 2013)
5. Burkhart, M., Dimitropoulos, X.: Fast privacy-preserving top-k queries using secret sharing. In: Proceedings of 19th International Conference on Computer Communications and Networks (ICCCN). pp. 1–7 (2010)
6. Clifton, C., Kantarcioglu, M., Lin, X., Vaidya, J., Zhu, M.: Tools for privacy preserving distributed data mining. SIGKDD Explorations 4(2), 28–34 (Jan 2003)
7. Elmehdwi, Y., Samanthula, B.K., Jiang, W.: Secure k-nearest neighbor query over encrypted data in outsourced environment. In: The 30th International Conference on Data Engineering. pp. 664–675. Chicago, USA (March 2014)
8. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., Tan, K.L.: Private queries in location based services: anonymizers are not necessary. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 121–132. ACM (2008)
9. Henecka, W., Schneider, T.: Faster secure two-party computation with less memory. In: Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security. pp. 437–446. ACM (2013)
10. Kaleli, C., Polat, H.: Privacy-preserving som-based recommendations on horizontally distributed data. Knowledge-Based Systems 33, 124–135 (2012)
11. Kantarcıolu, M., Clifton, C.: Privately computing a distributed k-nn classifier. In: European conference on principles of data mining and knowledge discovery. pp. 279–290. Springer (2004)
12. Qi, Y., Atallah, M.J.: Efficient privacy-preserving k-nearest neighbor search. In: Distributed Computing Systems, 2008. ICDCS'08. The 28th International Conference on. pp. 311–319. IEEE (2008)
13. Shaneck, M., Kim, Y., Kumar, V.: Privacy preserving nearest neighbor search. In: Machine Learning in Cyber Trust, pp. 247–276. Springer (2009)
14. Vaidya, J., Clifton, C.: Privacy-preserving kth element score over vertically partitioned data. IEEE Transactions on Knowledge and Data Engineering 21, 253–258 (Feb 2009)
15. Vaidya, J., Clifton, C., Zhu, M.: Privacy Preserving Data Mining, Advances in Information Security, vol. 19. Springer (2006)
16. Wong, W.K., Cheung, D.W.l., Kao, B., Mamoulis, N.: Secure knn computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. pp. 139–152. ACM (2009)
17. Wu, X., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Philip, S.Y., et al.: Top 10 algorithms in data mining. Knowledge and information systems 14(1), 1–37 (2008)
18. Xiong, L., Chitti, S., Liu, L.: Mining multiple private databases using a knn classifier. In: Proceedings of the 2007 ACM symposium on Applied computing. pp. 435–440. ACM (2007)
19. Yao, A.C.: Protocols for secure computation. In: Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science. pp. 160–164 (1982)
20. Zhang, F., Zhao, G., Xing, T.: Privacy-preserving distributed k-nearest neighbor mining on horizontally partitioned multi-party data. In: International Conference on Advanced Data Mining and Applications. pp. 755–762. Springer (2009)